LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Visualization Of Experimental Data At The National Ignition Facility

M. S. Hutton, R. Bettenhausen, E. Bond, A. Casey, R. Fallejo, J. Liebman, A. Marsh, T. Pannell, S. Reisdorf, A. Warrick

October 1, 2013

VISUALIZATION OF EXPERIMENTAL DATA
AT THE NATIONAL IGNITION FACILITY

Matthew S. Hutton, Rita Bettenhausen, Essex Bond, Allan Casey, Robert Fallejo, Judith Liebman, Amber Marsh, Thomas Pannell, Scott Reisdorf, Abbie Warrick, LLNL, Livermore, CA, 94550, U.S.A

Abstract

An experiment on the National Ignition Facility (NIF) may produce hundreds of gigabytes of target diagnostic data. Raw and analyzed data are accumulated into the NIF Archive database. The Shot Data Systems team provides alternatives for accessing data including a web-based data visualization tool, a virtual file system for programmatic data access, a macro language for data integration, and a Wiki to support collaboration. The data visualization application in particular adapts dashboard user-interface design patterns popularized by the business intelligence software community. The dashboard canvas provides the ability to rapidly assemble tailored views of data directly from the NIF archive. This design has proven capable of satisfying most new visualization requirements in near real-time. The separate file system and macro feature-set support direct data access from a scientist's computer using scientific languages such as IDL, Matlab and Mathematica. Underlying all these capabilities is a shared set of web services that provide APIs and transformation routines to the NIF Archive. The overall software architecture will be presented with an emphasis on data visualization.

INTRODUCTION

A target shot on the National Ignition Facility often results in the capture of hundreds of gigabytes of laser and target data. This data is collected via a cluster of software agents where it is transformed and loaded (ETL) into the NIF Archive.

Subsequent data analysis is performed either online via automation or offline by scientists. For example, the Shot Analysis and Visualization (SAVI) system is an analysis cluster that converts raw signal data into calibration-corrected, diagnostic-level measures such as neutron yield and hohlraum temperature. These measurements are typically available in the archive within 15 to 30 minutes of the shot.

During this timeframe, operations and scientists use data visualization to validate diagnostic performance and gain perspective on the outcome of the experiment.

Subsequently, the data visualization tool is used by scientists to explore data, correlate results, support experimental analysis and modelling, and for collaboration among scientific working groups.

Modelling and experimental analysis performed by scientists requires additional technologies that can deliver bulk data acquisition, data integration, and support for uploading of results back to the archive.

Data Visualization at the National Ignition Facility has radically evolved over the past three years with many lessons learned and at least one major technology reboot. In this document, we discuss key lessons learned and the evolution of complementary solutions that support the data requirements of our scientific user base.

First Data Visualization Attempt

NIF was commissioned in 2010 and new target diagnostics were getting simultaneously fielded. The grand challenge for data visualization was to develop a new software platform amidst undefined and rapidly changing user requirements. Each diagnostic team had unique preferences for how diagnostic data should be presented. The development team was overwhelmed with the sheer volume and variability of data required to satisfy these requirements.

The data visualization team initially used a conventional software engineering process with the goal of delivering new updates in three to four week delivery cycles.

The development team selected a Java Server Faces (JSF) framework to provide the type of rich, platform-independent Web 2.0 experience that users had begun to expect with the web at the time. JSF implementations provide pre-built, re-usable UI components that could reduce development times.

In practice, the team struggled to obtain the promised benefits with a JSF framework. While the ability to use pre-built JSF components could allow a developer to quickly wire together an application user interface, the pre-built JSF components required by NIF's visualization were rarely available. The team found themselves constructing components from scratch, a particularly expensive task. Furthermore, re-use became an elusive target.

The cyclic delivery schedule although short by some standards, was much too long for end-users. It seemed users needed to see their requirements implemented in order to better define them. This meant multiple cycles (months) before user requirements could be fully satisfied.

Additionally, performance of the new visualization tool was inadequate. Navigation was painfully slow as users sometimes needed to wait for the rendering of a page in transit to another page. The very nature of JSF page rendering relied on the server pre-aggregating data before it was sent to the browser for rendering. Efficiently querying the archive to retrieve hundreds of disparate records while the user waited proved to be an ongoing technical challenge. Attempts to improve performance led to complex SQL requiring a multi-disciplined team to

optimize and re-optimize queries as execution plans changed. In the end, overall performance was still slow for most pages. Users spent far too long watching a spinning progress meter.

Archive Viewer Background

A separate tool, the *Archive Viewer,* was established early-on as a primitive catch-all visualization capability for the archive. Utilizing the archive's metadata structures (discussed), the tool could render any content from the archive in a web-based user interface. It provided flexible search capabilities and data downloads in various formats. Expert users used the tool primarily as a back-door for data exploration and targeted downloads. Although it was fast and functional, it lacked the highly-tailored visualization desired by the users and presented by the JSF-based visualization tool.

In essence, NIF had two different visualization applications. One product required no ongoing development and the other struggled with the challenges mentioned earlier.

There was certain elegance and appeal to the "free" visualization provided by the Archive Viewer. It took advantage of key features of the archive such as introspection and metadata. However, it lacked the "gloss" demanded by users.

Visualization Redux - Dashboards

The advent of new, powerful JavaScript frameworks such as jQuery presented opportunities to rethink our architecture. Could we leverage the 'free visualization' aspects enabled by the archive, yet present a rich and tailored visualization experience to the user?

This objective in mind, a prototyping effort eventually led to a new d*ashboard* capability within the Archive Viewer. The success of this new approach quickly led to the de-commissioning of the old JSF-based visualization project.

In business, a dashboard consolidates key operational or performance metrics into a single, user-friendly page [1]. This is similar to what scientists and diagnostic teams required, data tailored and consolidated into a small set of pages that maximize information at-a-glance.

The Archive Viewer dashboard is created by choosing a layout, typically a grid. Pages are designed by assembling *widgets*. A widget is a self-contained component designed to render a particular type of content. For example, a widget might present a table of data, an image, or an interactive plot. The widget is *drag-and-dropped* from a palette into the layout. Over time, we designed widgets to become highly configurable so that a few dozen distinct widget-types could in combination, satisfy almost any arbitrary visualization need.
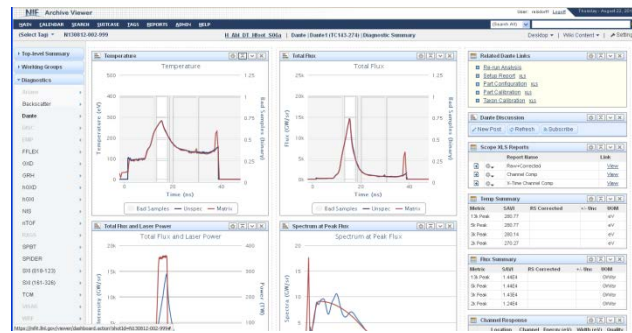

Figure 1: The NIF Archive Viewer Dashboard

The dashboard approach simultaneously solved multiple problems. Software developers were suddenly removed from the process of writing custom software code to present data. Secondly, performance from the perspective of the user radically improved. Each widget can take advantage of a browser thread and render itself independently. This gives the appearance of rapid application response as data begins to appear on the screen immediately.

In the first month of deployment we were able replicate nearly two years worth of custom development in the JSF-based visualization tool through dashboards. From there, the tools content diverged as new dashboards quickly emerged. Over several months we designed more flexible layouts and widget controls. Turnaround time to produce new dashboards (completely tailored views of data) shrunk from days to hours to near real-time. It is now common to craft a dashboard alongside the customer as she talks through her requirements.

Dashboards are typically defined around hierarchically-organized subject-areas. For example, each diagnostic is defined as a group of dashboards with summary and additional levels of detail. Some dashboards revolve around collaborative working groups such as symmetry, backscatter, or neutronics.

In addition to "single shot" dashboards, we have several dozen multi-shot dashboards which provide interactive trending across shots and comparisons to simulations. A dashboard can be made public or private (owned by an individual). Private dashboards allow us to provide even more tailored content to individuals without cluttering the menu.

Dashboard Architecture

The web-based dashboard front-end of the Archive Viewer is custom-written using jQuery and JavaScript MVC. These technologies led to an elegant and maintainable object-oriented, JavaScript dashboard framework. Open-source JavaScript charting frameworks such as HighCharts provide interactive plotting capabilities with series selection, auto-scaling, zooming, etc.
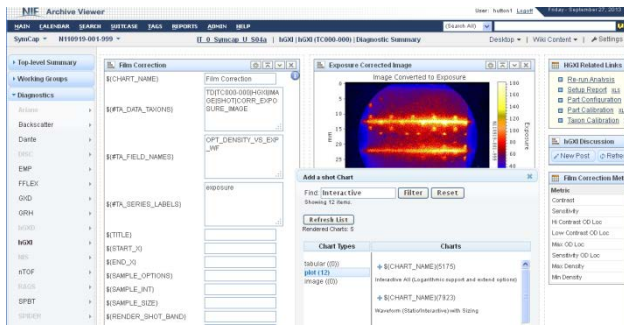
Figure 2: Creating a new dashboard in real-time. Widgets (plots, images, and tables) are dropped from a palette into a layout. The newly dropped widget (left) allows the user to configure data access bindings and plotting behaviour.

The NIF Archive

The NIF Archive itself enables many of our data visualization capabilities. The archive is the permanent repository for experimental data generated by the NIF. In addition to capturing laser and target diagnostic data, the archive contains setup parameters, serialized-part history, instrument calibration, etc. This data is essential for performing data analysis and establishing provenance of experimental results. Data visualization uses the archive as its sole data source.

The archive stores data in an Oracle 11g RAC database cluster. We determined that most data is structured and therefore scalar components are organized into tables and columns making trending possible. Waveforms and images are transformed into a NIF canonical layout, packed into a Hierarchical Data Format (HDF) document and stored as Binary Large Objects (BLOBS) in the database.

The archive provides an important *enrichment* function that simplifies the visualization process. An asynchronous enrichment agent generates JPEG display images of various sizes and color maps immediately following archival. These images are stored as metadata alongside the original HDF image or waveform. The ability to quickly display web-friendly images dramatically improves visualization performance.

The archive's data structures are also fundamental to the visualization process. The archive is built on top of a low-level Content Management framework. The content management core provides important features such as versioning, data access control, and metadata. It enabled us to logically model our data structures into object-oriented classes. Although data is stored relationally in the database it is represented by the API as hierarchical objects that can be introspected by visualization codes. These abstractions form one of the key building blocks that enable the visualization engine.

The archive makes extensive use of metadata useful in data visualization. One primary example of metadata is the NIF *data taxon*. Every datum is classified by a taxonomic label. This multi-part key is used extensively to relate data to its origin, diagnostic family, and the analysis process that generated it. Other examples of metadata include data quality, class and attribute descriptions, user comments, and data provenance (ie. references to all data and algorithms used to generate a processed result).

The Archive API is typically accessed by clients through a SOAP-based Web Service. The web service provides an abstraction layer which enabled development of multiple data visualization and integration solutions described in this document.

WebDAV

The *Archive WebDAV* server provides a web-based filesystem alternative for accessing the NIF archive. Web Distributed Authoring and Versioning (WebDAV) is a standards-based extension to HTTP that can support document collaboration [2]. Initially, all document-oriented data such as waveforms, images (raw or for display purposes) are served from WebDAV.

The Archive Viewer provides a convenient method for browsing data visually; but often scientists need to download large sets of data after each shot. Requiring users to navigate web pages and click to download each item quickly becomes a chore.

In 2012, WebDAV was enhanced to address the needs of bulk downloads and automation. We use WebDAV to offer a virtual file system to the entire archive. All the data in the archive is presented as if they were files on the file system. As such, WebDAV can provide a predictable URL (path) to data across shots.

The file system is considered virtual because data is actually maintained in the database. Content is rendered only when the user actually downloads a file. This also allows WebDAV to present data in many alternate formats. These formats are rendered at request time requiring no additional copies of data. For example, an HDF image can be rendered as a PNG, TIF, or a JPG file. Waveforms and scalar data can be HDF or various plain-text formats.
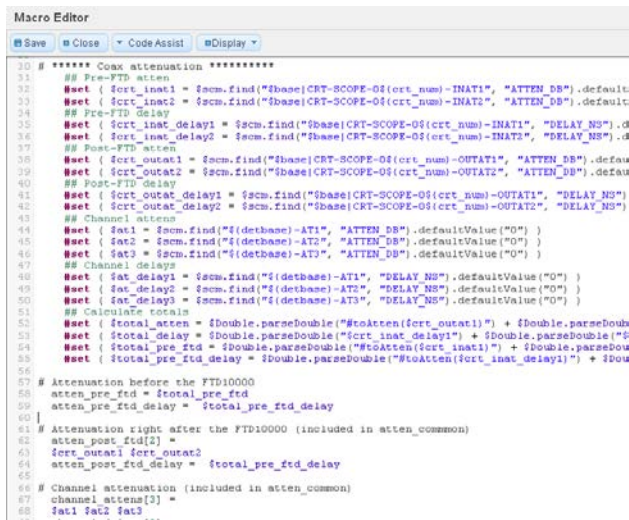
WebDAV as an HTTP-based protocol can be navigated with a simple web browser or be mounted by most modern operating systems as a networked drive. It provides a lowest-common-dominator for users, as almost any language or tool can query HTTP content. Examples of popular tools include Unix wget and curl, desktop analysis languages such as IDL, MATLAB, and Mathematica, and general languages such as Java, Perl, and Python. WebDAV has become a powerful medium for automating user data downloads.

Desktop Analysis and Macros

By delivering predictable paths (URLs) for fetching data from WebDAV, users can script simple bulk downloads to their desktop. However, most sophisticated desktop analysis involves fetching dozens or even hundreds of individual files, parsing them to acquire specific scalar values, and potentially performing conditional lookups to resolve calibration. The data acquisition and consolidation codes could easily exceed the algorithmic code required to analyze the data.

Building upon WebDAV and the Archive Web Services layer, we introduced a Domain Specific Language (DSL) enabling user-written macros. A DSL is language tailored to a particular problem domain [3]. The macro assembles disparate scalar information into a single text file using concise, user-friendly language syntax.

The macro consolidates all values into a single property file of the user's design. It also collects binary files such as images into a single folder making the job of scripting as simple as scooping a set of files from a directory. In addition, it provides a WebDAV folder for the user to upload analysis results back into the archive. Using macros, the chore of writing data acquisition codes for an end-to-end diagnostic desktop analysis can be written very quickly, typically in a few hours.

Figure 3: A user-defined macro defines all inputs required to perform NTOF diagnostic analysis on the desktop. The macro language allows the user to layout the input file and substitute macro mark-up for parameters.

Figure 4: A sample of the macro-generated parameter file for NTOF. The macro is not executed until the user attempts to access the file from WebDAV.

The macro capability has proven powerful enough to become a key component in the new data integration engine used by the SAVI analysis system, a software cluster that performs automated analysis of target diagnostic data on the NIF [4].

Finally, we make available WebDAV client libraries for popular scientific analysis tools such as IDL and Mathematica. This allows these tools to quickly connect and authenticate with WebDAV. The libraries also provide utilities for efficiently working with NIF canonical data formats.

Self-service

We continue to evolve towards self-sufficiency. Software developers can focus on frameworks that provide general capabilities rather than writing reports or servicing ad-hoc data requests.

As tools mature and the collective knowledge-base improves, we see users capable of developing their own dashboards, writing their own macros to acquire data from the archive, and even integrating their own codes into the automated processing cluster provided by SAVI.

Indeed, users are beginning to do these things and as a result, a more technical and self-sufficient culture is emerging among our users.

Summary

Meeting the needs of scientific users was a learning process requiring willingness to engage new technologies and abandon failed efforts. A good data visualization solution must be able to rapidly deliver against requirements, preferably in real-time. New visualization must be able to be accomplished outside of typical software delivery schedules. Performance is always a first-class concern.

The web-based dashboard model provides an effective visualization solution for NIF. However, good data visualization capabilities are not enough. As scientists need to analyze data with their own codes, streamlining data provisioning is also a priority. Higher-level services such as user-defined macros and client libraries for popular tools provide a comprehensive data integration solution for scientists.

REFERENCES

[1] "Dashboard (business)", Wikipedia: The Free Encyclopedia; http://en.wikipedia.org/wiki/Dashboardbusiness

[2] "WebDAV"; Wikipedia: The Free Encyclopedia; http://en.wikipedia.org/wiki/WebDAV

[3] "Domain-specific language"; Wikipedia: The Free Encyclopedia; http://en.wikipedia.org/wiki/Domain-specific_language

[4] A. Casey, et al, "Flexible Data Driven Experimental Data Analysis at the National Ignition Facility", ICALEPCS'13, TUPPC072